Digital Filter Synthesis Based on Minimal Signed Digit Representation

In-Cheol Park

Dept. of Electrical Engineering and Computer Science
2nd Floor, Chips Building, EE, KAIST,
373-1 Guseong-dong Yuseong-gu Taejeon, Korea
+82-42-869-3461
icpark@ee.kaist.ac.kr

Hyeong-Ju Kang

Dept. of Electrical Engineering and Computer Science
2nd Floor, Chips Building, EE, KAIST,
373-1 Guseong-dong Yuseong-gu Taejeon, Korea
+82-42-869-8061
dk@ics.kaist.ac.kr

ABSTRACT

As the complexity of digital filters is dominated by the number of multiplications, many works have focused on minimizing the complexity of multiplier blocks that compute the constant coefficient multiplications required in filters. The complexity of multiplier blocks can be significantly reduced by using an efficient number system. Although the canonical signed digit representation is commonly used as it guarantees the minimal number of additions for a constant multiplication, we propose in this paper a digital filter synthesis algorithm that is based on the minimal signed digit (MSD) representation. The MSD representation is attractive because it provides a number of forms that have the minimal number of non-zero digits for a constant. This redundancy can lead to efficient filters if a proper MSD representation is selected for each constant. In experimental results, the proposed algorithm resulted in superior filters to those generated from the CSD representation.

1. INTRODUCTION

Digital filters are frequently used in digital signal processing by virtue of stability and easy implementation. Although programmable filters based on digital signal processing cores can take an advantage of flexibility, they are not suitable for recent consumer applications demanding high throughput and low power consumption. In such an application, therefore application specific digital filters are frequently adopted to meet the constraints of performance and power consumption. Number systems have a great influence on the hardware complexity of digital filters. Although the two's complement representation is commonly used for digital computation, the signed digit representation is frequently more efficient in digital filters where signal values are multiplied by many constant coefficients.

The problem of designing digital filters has received a great attention during the last decade, as the filters are suffering from a large number of multiplications, leading to excessive area and power consumption even if implemented in full custom integrated circuits. Early works have focused on replacing multiplications by decomposing them into simple operations such as addition, subtraction and shift. As the coefficients of an application specific filter are constant, the decomposition is more efficient than employing general multipliers. The number of additions/subtractions used to implement the coefficient multiplications in this case dominates the complexity of filters. To reduce the complexity, the coefficients can be restricted to powers-of-two or expressed in canonical signed digit (CSD) representation to minimize the number of additions/subtractions required in each coefficient multiplication. On average, the CSD representation can reduce 33% of non-zero digits compared with the binary representation. To further reduce hardware complexity in the applications requiring multiple constant multiplications (MCMs), a common subexpression is searched among multiple constants and implemented into one hardware block in order to share the result of the subexpression in evaluating all the constants. Many approaches have been proposed for the implementation of MCMs [1],[2],[3],[6],[8],[10],[11],[12],[14],[15]. Previous approaches have tried to select common subexpressions as many as possible after representing the constants in the CSD representation. Although the CSD representation is good for one constant, it is not the best for multiple constants because the CSD representation of a constant is unique and independent of the other constants, leading to limited subexpressions for multiple constants. For the multiple constant multiplications, it is more efficient to use the minimal signed digit (MSD) representation which has the same number of non-zero digits as the CSD representation but provides multiple representations for a constant [8],[15].

As the CSD representation is mathematically unique, it has received much attention and there have been many methods of converting a given binary number into the CSD representation [5],[7],[9],[13],[16]. The uniqueness is significant in developing algorithms, but not in implementing hardware units. In general, the MSD representation providing multiple representations that yield the same value is more flexible than the CSD representation. This redundancy can result in smaller hardware units than those generated from CSD representation, if an appropriate MSD representation is selected for each constant. In spite of this advantage, the MSD number system has not been studied much

Table 1. Examples of MSD Representations.

171	$0101010\bar{1}0\bar{1}$	175	$0101010\bar{0}0\bar{1}$
	$0011010\bar{1}0\bar{1}$		$0011010\bar{0}0\bar{1}$
	$0010110\bar{1}0\bar{1}$		$0010110\bar{0}0\bar{1}$
	$0010101\bar{1}0\bar{1}$	176	$0101010\bar{0}00$
	0010101011		$0011010\bar{0}00$
172	$0101010\bar{1}00$		0010110000
	$0011010\bar{1}00$	177	$0101010\bar{0}01$
	$0010110\bar{1}00$		$0011010\bar{0}01$
	0010101100		0010110001
173	$0101010\bar{1}01$	178	$0101010\bar{0}10$
	$0011010\bar{1}01$		$0011010\bar{0}10$
	$0101010\bar{0}1\bar{1}$		0010110010
	$0010110\bar{1}01$	179	$0101010\bar{1}0\bar{1}$
	$0011010\bar{0}1\bar{1}$		$0011010\bar{1}0\bar{1}$
	0010101101		$0101001\bar{1}0\bar{1}$
	$0010110\bar{0}1\bar{1}$		0101010011
174	$0101010\bar{0}10$		$0010110\bar{1}0\bar{1}$
	$0011010\bar{0}10$		$0011001\bar{1}0\bar{1}$
	$0010110\bar{0}10$		$0011010\bar{0}11$
			0010110011

because of the lack of formalism and there has been no report on how to find all the MSD representations for a given number except enumerating all cases. Recently, we developed an efficient algorithm that can search all the MSD representations for a number. Based on the algorithm we propose in this paper a new digital filter synthesis algorithm that can exploit the redundancy of MSD representation. Experimental results show that the proposed algorithm provides more efficient multiplier blocks than those generated from the CSD representation.

The rest of this paper is organized as follows. In Section 2, the CSD representation and the MSD representation are introduced and compared in more detail. In Section 3, we summarize the MSD generation algorithm and the related theorems. The proposed filter synthesis algorithm is explained in Section 4, and experimental results obtained from several filter examples are described in Section 5. Finally, concluding remarks are made in Section 6.

2. CSD AND MSD REPRESENTATIONS

In this section, the CSD representation and the MSD representation are explained and compared in terms of multiplier block synthesis, and then the problem to be solved is defined.

CSD Representation: The CSD representation is a radix-2 signed digit system with the digit set $\{1,0,\bar{1}\}$, where $\bar{1}$ denotes -1. Given a constant, the corresponding CSD representation is unique and has two properties; the first is that the number of non-zero digits is minimal and the second is that the product of adjacent two digits is zero, that is, two non-zero digits are not adjacent. Due to the first property, the CSD representation is widely used in implementing MCMs because it guarantees the least number of additions for a given constant multiplication. The second property is called "property M" in [5]. If a signed digit representation of a constant satisfies property M, it is the CSD representation.

MSD Representation: If the second property is relaxed in the CSD representation, it is called minimal signed digit (MSD) representation. As shown in Table 1 that includes the MSD representations for constants ranging from 171 to 179, a constant has not a unique MSD representation but many MSD representations usually.

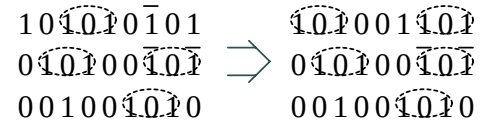

Figure 1. Pattern extraction with CSD and MSD representations.

Although the CSD representation is optimal for one constant, it is difficult to consider the other constants in case of multiple constants because a number is uniquely represented in CSD representation. Since the MSD representation is a super set of the CSD representation and provides a number of forms, the MSD representation is more appropriate in finding common subexpressions for multiple constants if a proper MSD form is selected for each constant to be synthesized [8],[15]. Since the representation method affects the number of additions (or subtractions) in the decomposed multiplication block and the number of common subexpressions that can be eliminated, it has significant influence on the resulting area and power consumption.

The most important step in solving the MCM problem is to extract the most common patterns. An algorithm was proposed to search the patterns in [12], but only the CSD representation was considered there as shown in the left side of Figure 1. The CSD representation is used because it has the minimum number of non-zero digits. To cope with the demerit of the CSD representation that it cannot exploit the flexibility of signed digit representation, the MSD representation that provides multiple forms for a constant can be applied. If each constant is represented with a proper MSD form, more patterns can be extracted as shown in the right side of Figure 1. In Figure 1, we regard $\bar{1}0\bar{1}$ as the same pattern of 101 because 101 can be evaluated by $-(101)$.

The problem to be solved is described as follows:

> *Problem. Given a set of filter coefficients, generate a multiplier block that requires the minimal number of adders/subtractors.*

In this paper, the delay is specified by the number of adder-steps that denotes the maximal number of adders/subtractors to be passed through to produce any multiplication. For a set of coefficients, $c_1, c_2, \cdots, c_m$, the low bound of adder-steps, N, required in implementing the multiplier block is given by $N = \max\{\lceil \log_2 k_i \rceil\}$, where k_i is the number of non-zero digits in the CSD format of c_i. The equality holds when each multiplication is constructed by using a complete binary tree of adders. One simple method of achieving N is to construct coefficients individually by using a separate binary tree of adders for each c_i, meaning that adders associated with c_i are not shared with those of other c_j's.

3. MSD GENERATION ALGORITHM

In this section, we present an algorithm to generate all the MSD

representation of a number. Basically, the MSD representations are derived from the corresponding CSD representation that can be obtained using the algorithm in [5]. We summarize only the main theorems for the completeness of this paper. For a number N, the CSD reperesentation and the MSD representation are represented as $c_{n-1}c_{n-2}\cdots c_0$ and $m_{n-1}m_{n-2}\cdots m_0$, respectively, where the digit set is $\{1,0,\bar{1}\}$ in both representations.

Definition 1. Given a sequence of $c_{n-1}c_{n-2}\cdots c_0$, $(c_i = \pm 1,0)$, *the sequence possesses "property M" if* $c_i c_{i-1} = 0$, *for* $1 \le i \le n-1$.

Theorem 1. [5] Among all the sequences of c_i*'s which yield N, there is only one sequence that meets property M and the unique sequence has the minimum number of non-zero digits.*

The unique representation of N that has the property M is usually called the CSD representation. If there is only one MSD representation, it is equivalent to the CSD representation. However, since we are now searching for other MSD representations, we assume hereafter that there is an MSD representation, $m_{n-1}m_{n-2}\cdots m_0$, that is different from the CSD representation, $c_{n-1}c_{n-2}\cdots c_0$.

Theorem 2. Assume that a number N has the CSD representation $c_{n-1}c_{n-2}\cdots c_0$ *and an MSD representation* $m_{n-1}m_{n-2}\cdots m_0$. *If there is a portion ranging from k to l,* $0 \le l < k \le n-1$ *which satisfies the followings,*

$$c_{k+1} = m_{k+1},\ c_k \ne m_k,\ c_{k-1} \ne m_{k-1},\ \cdots,\ c_l \ne m_l,\ c_{l-1} = m_{l-1}$$

then $k-l$ *is even and*

$$c_k c_{k-1}\cdots c_l = 10\bar{1}0\bar{1}\cdots\bar{1}0\bar{1} = 1(0\bar{1})^+$$
$$m_k m_{k-1}\cdots m_l = 01010\cdots011 = (01)^+1,$$

or

$$c_k c_{k-1}\cdots c_l = \bar{1}0101\cdots101 = \bar{1}(01)^+$$
$$m_k m_{k-1}\cdots m_l = 0\bar{1}0\bar{1}0\cdots0\bar{1}\,\bar{1} = (0\bar{1})^+\bar{1}.$$

If $k = n-1$, we can make the above case by expanding a digit, $c_n = m_n = 0$. Similarly, we expand a digit, $c_{-1} = m_{-1} = 0$, if $l = 0$.

Corollary. Given a sequence of c_i*'s such as* $1(0\bar{1})^+$ *or* $\bar{1}(01)^+$, *the number of MSD representations that are different from the sequence is equal to the number of zero digits in the sequence.*

This corollary gives a hint to count the number of MSD representations. For the CSD representation of N, let the number of such maximal sequences be t and the number of zero digits in sequence i be Z_i. The maximally possible number of MSD representations, including the CSD representation, is given by $\prod_{i=0}^{t-1}(Z_i+1)$. The number is obtained with assuming that the sequences are disjoint. However, in practice, two sequences can be overlapped with each other, as shown in Figure 2(a). A sequence can be overlapped with two other sequences, an upper one and a lower one. Since a sequence restricts the other sequences overlapped, we have to subtract the cases in counting the number of MSD representations. If each MSD representation of a sequence is denoted as a node and an edge is drawn if two nodes can be applied simultaneously, we can obtain a graph, as shown in Figure 2(b). In the graph, each path from a node of the leftmost sequence to a node of the rightmost sequence corresponds to an MSD representation of N. Therefore the total number of MSD representations can be obtained by counting the number of paths. The dynamic programming [4] can be used to count the total number of paths, which results in a recursive equation given below.

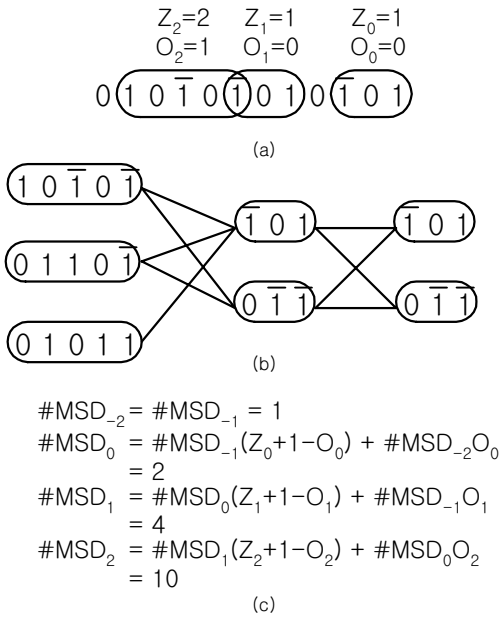

Figure 2. Counting the total number of MSD representations. a) An example that has overlapped sequences. b) A graph model to count the number. c) Counting procedure for the graph.

$$MSD_{-2} = MSD_{-1} = 1,$$
$$MSD_i = MSD_{i-1}(Z_i + 1 - O_i) + MSD_{i-2}O_i$$

where O_i is 1 if sequence i is overlapped with sequence $i-1$, otherwise 0.

From Theorem 2, it is induced that the only transformations needed to convert the CSD representation to MSD representations are $10\bar{1} \to 011$ and $\bar{1}01 \to 0\bar{1}\,\bar{1}$. Figure 3 shows how a number of MSD representations are achieved by repeatedly applying the short transformations. The overall algorithm

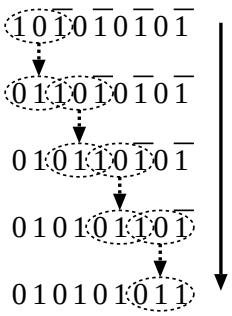

Figure 3. Decomposition of a long conversion into several short conversions.

developed from that fact is as follows. First, a number is represented in CSD using one of the algorithms presented in [5],[7],[9],[16]. As the CSD representation is also an MSD representation, it is registered as the first MSD representation. Next, a pattern of either $10\bar{1}$ or $\bar{1}01$ is searched starting from the most significant digit and transformed to 011 or $0\bar{1}\bar{1}$ respectively. For each transformation, a new MSD representation is generated. The transformation is repeatedly applied to the new MSD representations found in the previous transformations until there is no such a pattern. To avoid duplications, the pattern is searched in an MSD representation from the next position of the digit where a transformation is applied to generate the MSD representation. The detailed explanation of the algorithm and related terms are described below.

Definitions

N : an n bit number to find all MSD representations.

MSD_i : the *i*-th MSD representation found.

S : a set including the MSD representations found.

|S| : the number of MSD representations in *S*.

SearchMSD : the MSD representation where a new one is being searched.

SP[i] : the digit position where the transformation is applied to generate *i*-th MSD representation.

SearchPoint : the digit position where the search is being done.

The algorithm

Step 1. Convert *N* into the CSD presentation. It is named MSD_0. $S = \{MSD_0\}$. $|S| = 1$. *SearchMSD* = 0. *SP[0]* = n-1.

Step 2. *SearchPoint* = *SP[SearchMSD]*.

Step 3. If *SearchPoint* < *1*, go to Step 6.

Step 4. If the digits from position *SearchPoint* to *SearchPoint-2* in $MSD_{SearchMSD}$ are $10\bar{1}$ or $\bar{1}01$, make a new MSD by changing $10\bar{1}$ to 011 or $\bar{1}01$ to $0\bar{1}\bar{1}$, respectively. The new MSD is named $MSD_{|S|}$. *SP[NumMSD] = SearchPoint-2*. Insert the new MSD into *S*. Increment |*S*|. Decrement *SearchPoint* by 2. Go to Step 3.

Step 5. Decrement *SearchPoint*. Go to Step 3.

Step 6. Increment *SearchMSD*. If *SearchMSD* is the same as |*S*|, end. Otherwise, go to Step 2.

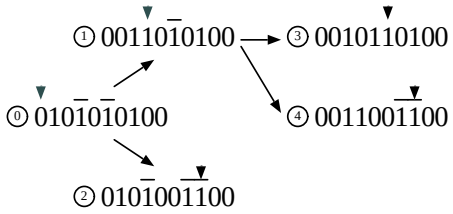

Figure 4. The MSD generation procedure for 180.

As an example, Figure 4 shows the procedure of finding all the MSD representations of 180. The circled number means the order of MSD representations generated by the proposed algorithm. The first MSD representation numbered as 0 is equivalent to the CSD representation of 180. The inverse triangle of each MSD representation denotes the first *SearchPoint, SP[i]* of the representation.

4. DIGITAL FILTER SYNTHESIS ALGORITHM

In this section, we explain the proposed multiplier block synthesis algorithm. Before starting the explanation, we briefly introduce the previous filter synthesis algorithm proposed by Hartley [12], which is based on the CSD representation. The algorithm is selected for comparison, because it produces the best results among a number of CSD-based algorithms [10],[11],[12],[14].

Hartley's method is to combine sub-expressions that are common to multiple coefficients. The first step is to convert the coefficients into CSD representations, as the CSD representation guarantees the minimal number of non-zero digits for each coefficient. And then it searches a pair of two non-zero digits that are the most common among all possible pairs.

The selected pair is removed from the corresponding CSD coefficients and replaced by a new identifier assigned to the pair. The first identifier is 2 and the next is 3. Beginning from 2, the identifier increases by one for the next selected pair. The selecting procedure is repeated until there is no common sub-expression. Although Hartley's method is effective in finding maximally common sub-expressions, it is a greedy algorithm that can be easily trapped into a local minimum. It is not easy to apply Hartley's algorithm to the MSD representation as the representation of each coefficient has to be determined before applying the algorithm, which is another difficult work to be solved.

In Hartley's method, common sub-expressions are searched and combined after all coefficients are expressed in CSD representation, whereas in the proposed algorithm coefficients are considered and synthesized one by one, e.g., a coefficient is selected and synthesized sequentially one at a time.

The first step is to generate all MSD representations for each coefficient. Then a coefficient is selected for synthesis, which can be implemented using a minimal number of adders, that is, a coefficient with the minimal number of non-zero digits is selected for the first synthesis. The intermediate sums that can be obtainable from the adders used for the synthesized coefficients are registered as partial sums. Among not-yet synthesized coefficients, we select a coefficient that can be implemented with minimal additional adders. Therefore the next coefficient to be

synthesized is the one that can be implemented with using previously defined partial sums. The following is the flow of our algorithm where *cSet* is the set of coefficients that are to be synthesized and *patSet* is the set of patterns of partial sums used to synthesize the selected coefficient.

Step 1. Even coefficients are made odd by dividing by a power of 2. Negative coefficients are converted to positive ones. The coefficients that have the same value with another coefficients are removed. The remaining coefficients are inserted into *cSet*.

Step 2. Obtain all CSD and MSD representations of the elements in *cSet*.

Step 3. Insert 1 into *patSe*t.

Step 4. If there is an element in *cSet* that has the same MSD representation with shifted value of an element in *patSet*, it is removed from *cSet*. If no element is in *cSet*, end. Repeat this step until there is no such an element in *cSet*.

Step 5. If there is an element in *cSet* that has the same MSD representation with a shifted combination of two elements in *patSet*, it is removed and inserted into *patSet*. If no element is in *cSet*, end. Repeat this step until there is no such an element in *cSet*.

Step 6. If there is an element in *cSet* that has the same MSD representation with a shifted combination of three elements in *patSet*, it is removed and inserted into *patSet*. One combination of two partial sums is inserted into *patSet*. If no element is remained in *cSet*, end. If no element is removed from *cSet* in Steps 4, 5, and 6, go to Step 7. Otherwise, go to Step 4.

Step 7. For each element in *cSet*, make a shifted combination of two partial sums in *patSet*, and check if the pattern of the shifted combination is included in an MSD representation of the element. We select a combination that maximally matches to an MSD representation. After doing this for all remaining elements, we select an element which has the most matched combination of two partial sums. The pattern of the combination is registered as a new partial sum. A new element obtained by removing the selected combination is inserted into *cSet*. Go to Step 4.

When combining elements, shifting of elements is allowed, but there must be no conflict between them, i.e., no digit place where two or more elements have non-zero digits simultaneously is allowed. Step 1 is the preparation of our algorithm. The division or multiplication by a power of 2 can be implemented by wiring and the negation can be implemented by replacing the adders with the subtractors. We can use positive, odd numbers in place of negative or even numbers, and the original negative or even coefficients can be produced from the positive, odd numbers with a little hardware overhead. Step 2 generates the CSD representation and all of the MSD representations for each coefficient. The MSD representations are created with the algorithm in Section 3. In Step 3, 1 is inserted into *patSet* because it needs no adder. In Step 4, the elements in *cSet* that are already in *patSet* are removed because they are already made. In Step 5 the elements that can be synthesized with only one adder are selected and synthesized. In Step 6, the elements that need two adders are synthesized. In Step 7 we modify a coefficient by including the most matched combination of partial sums into *patSet* when there is no element that can be synthesized with one or two adders.

Table 2. Test Filter Specification.

Filter	Passband	Stopband	#tap	Width
1	0.15	0.25	40	12
2	0.15	0.25	60	14
3	0.15	0.20	60	14
4	0.15	0.20	100	16
5	0.10	0.15	60	14
6	0.10	0.15	100	16
7	0.10	0.12	100	16
8	0.10	0.12	120	18

5. EXPERIMENTAL RESULTS

The proposed algorithms are applied to several FIR filters and compared with previous algorithms. The specification of those filters is summarized in Table 2, where f_p and f_s are normalized passband frequency and stopband frequency respectively, *#tap* is the number of taps, and *Width* is the word size in fixed-point integer representation. The coefficients of test filters are generated with the f_p, f_s, and *#tap* using the Remez algorithm in MATLAB and are converted to integer numbers with rounding. The passband and stopband frequency of the first filter in Table 2 are quoted from the example in [6]. We assume transposed-form filters because they can accept the subexpression sharing.

In Table 3, the results obtained by the previous and proposed algorithms are shown. The column denoted as simple represents the results obtained by constructing a separate adder tree for each coefficient. The simple method requires a lot of adders but provides the fastest results requiring the minimal number of adder-steps. The next two columns show the results of previous CSD-based algorithms. Among them, Hartley's method provides better results. Comparing to the simple method, Hartley's method significantly reduces the number of adders needed to synthesize the filters at the cost of a little increase of delay for some filters.

The next two columns describe the results obtained by applying the proposed algorithm. The MSD-based algorithm provides fast results that need the minimal number of adders. It can reduce the number of adders by 10% even compared to Hartley's method without increasing the number of adder-steps. The CSD-based results are included to show the effectiveness of the proposed algorithm. In this case, only the CSD representations are considered in the proposed algorithm. Although the algorithm is proposed for the MSD-representation, the CSD-based results are comparable with the Hartley's results. This implies the proposed algorithm is as effective as Hartley's algorithm for the CSD representation.

6. CONCLUSION

In this paper, we have presented a new digital filter synthesis algorithm that is based on the MSD representation. Starting from the CSD representation, all the MSD representations are discovered by repeatedly applying simple transforms, $10\bar{1} \rightarrow 011$ and $\bar{1}01 \rightarrow 0\bar{1}\bar{1}$. The proposed filter synthesis algorithm is to select one coefficient at a time and synthesize it using previously synthesized patterns, which is different from the conventional method that searches sub-expressions common to

Table 3. Experimental Results for FIR Filters.

Filter	Simple		Potknjak[11]		Hartley[13]		Proposed algorithm MSD-based		Proposed algorithm CSD-based	
	#adders	#adder-steps	#adders	#adder-steps	#adders	#adder-steps	#adders	#adder-steps	#adders	#adder-steps
1	33	3	22	4	19	3	16	3	17	3
2	54	3	30	4	25	3	23	4	24	3
3	77	3	48	5	35	3	35	3	37	3
4	131	3	82	4	55	4	51	4	55	4
5	88	3	54	4	37	4	34	4	38	4
6	140	3	87	5	55	4	50	4	52	5
7	173	3	95	5	71	4	70	4	83	4
8	246	3	136	6	96	4	91	4	104	5
Avg.	240%	-	141%	-	100%	-	94%	-	104%	-

multiple constants. To show the effectiveness of the proposed algorithm, several filters are synthesized and compared with those generated from previous algorithms. The experimental results show that the proposed algorithm yields better results than the conventional ones obtained from the CSD representation. In addition, when the proposed algorithm is restricted to the CSD representation in making multiplier blocks, the results are worse than those generated by allowing the MSD representation. This means the MSD representation is more appropriate in filter synthesis than the CSD representation.

7. ACKNOWLEDGMENTS

This work was supported (in part) by the Korea Science and Engineering Foundation through the MICROS center at KAIST, Korea.

8. REFERENCES

[1] A. G. Dempster and M. D. Macleod, "Use of minimum adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. Ⅱ, vol. 42, no. 9, pp. 569-577, 1995.

[2] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," IEE Proc. G, vol. 138, no. 3, pp. 401-12, 1991.

[3] D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in Proc. 1993 IEEE Int. Symp. on Circuits and Systmes, 1993, pp. 84-87.

[4] E. Horowitz and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, 1978.

[5] G. W. Reitweisner, "Binary arithmetic," in Advances in Computers, F. L. Alt, Ed., vol. 1, chapter 5, pp. 232-308. Academic Press, New York, 1960.

[6] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," IEEE Trans. Circuits Syst., vol. 36, no. 7, pp. 1044-1047, 1989.

[7] I. Koren, Computer Arithmetic Algorithms, Prentice-Hall, 1993.

[8] J. T. Kim, Design and implementation of computationally efficient FIR filters, and scalable VLSI architectures for discrete wavelet transform, Ph.D. thesis, Korea Advanced Institute of Science and Technology, 1998.

[9] K. Hwang, Computer Arithmetic Principles, Architecture, and Design, John Wiley and Sons, Inc., 1979.

[10] M.Potkonjak, M. B. Srivastava, and A. Chandrakasan, "Efficient substitution of multiple constant multiplication by shifts and additions using iterative pairwise matching," in Proc. 31st ACM/IEEE Design Automation Conf., 1994, pp. 189-194.

[11] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Synthesis of multiplier-less FIR filters with minimum number of additions," in Proc. IEEE/ACM Int. Conf. on Computer Aided Design, 1995, pp. 668-671.

[12] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. Ⅱ, vol. 43, no. 10, pp. 677-688, 1996.

[13] R. Hashemian, "A new method for conversion of a 2's complement to canonic signed digit number system and its representation," in Proc. Asilomar Conf. Signals, Syst., Computers, 1997, pp. 904-907.

[14] R. Paško, P. Schaumont, V. Derudder, S. Vernalde, and D. Ďuračková, "A new algorithm for elimination of common subexpressions," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 1, pp. 58-68, 1999.

[15] T. S. Chang, C. S. Kung, and C. W. Jen, " A simple processor core design for DCT/IDCT," IEEE Trans. Circuits Syst. Video Technol., vol. 10, no. 3, pp. 439-447, 2000

[16] Y. C. Lim, J. B. Evans, and B. Liu, "Decomposition of binary integers into signed power-of-two terms," IEEE Trans. Circuits Syst., vol. 38, no. 6, pp. 667-672, 1991.